Effective flowgraph-based malware variant detection

Silvio Cesare, Ph.D. Candidate, Deakin University
http://www.foocodechu.com
silvio.cesare@gmail.com

Who am I and where did this talk come from?

- Ph.D. Student at Deakin University.
- Research interests include:
  - Automated vulnerability discovery.
  - Software similarity and classification.
  - Malware detection.

- This presentation is based on my malware research.

Outline

1. Introduction (you might already know this)
2. New approaches to flowgraph-based classification
3. Evaluation
4. Other things we use our system on.
5. Conclusion

Introduction

This is to make sure everyone is up to speed.

If you've been to my presentations before, you might have already seen it.

INTRODUCTION

- Malware a significant problem.
- Static detection of malware a dominant real-time technique.
- Detecting unknown variants from known samples very useful.

Klez.a
Klez.b
Klez.c
Klez.d
...

Roron.ao
Roron.b
Roron.d
Roron.e
Roron.f
...

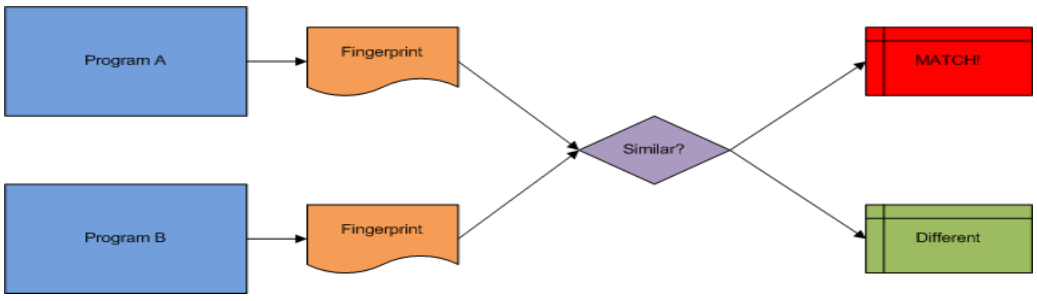
SIGNATURES AND BIRTHMARKS

- A birthmark is an invariant property in related samples.
- Birthmark comparison should allow inexact matching.

LIMITATIONS OF EXISTING BIRTHMARKS

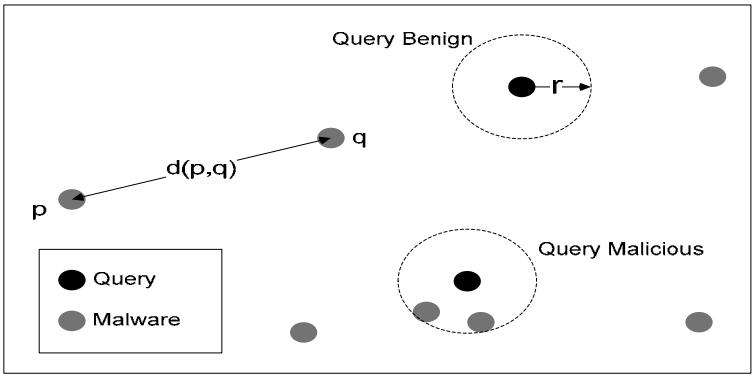
- Byte-level content can change in every variant.
- Comparing birthmarks often exact matching only.
- Inefficient for inexact database searching.
- Unable to detect unknown variants of known samples.
- Program structure a better birthmark.

THE SOFTWARE SIMILARITY PROBLEM



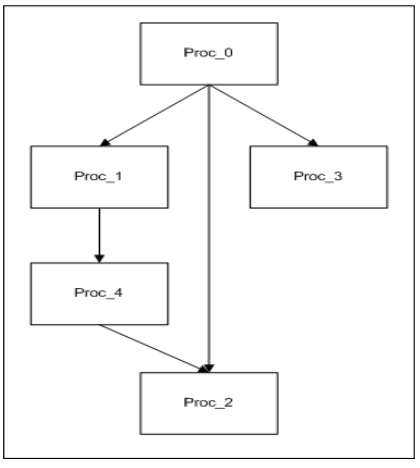
THE SOFTWARE SIMILARITY SEARCH

- Need a dissimilarity or distance metric.
- "Metric" property allows efficient database search.



EXISTING APPROACHES: A CALL GRAPH BIRTHMARK

- Inter-procedureal control flow.

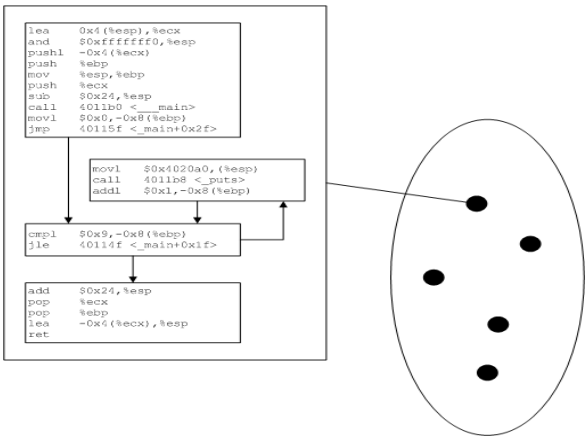


An optimal dissimilarity metric for graphs

- Graph edit distance.
- Number of operations to transform one graph to another.
- Complexity in NP.
- Non optimal solutions possible in cubic time.

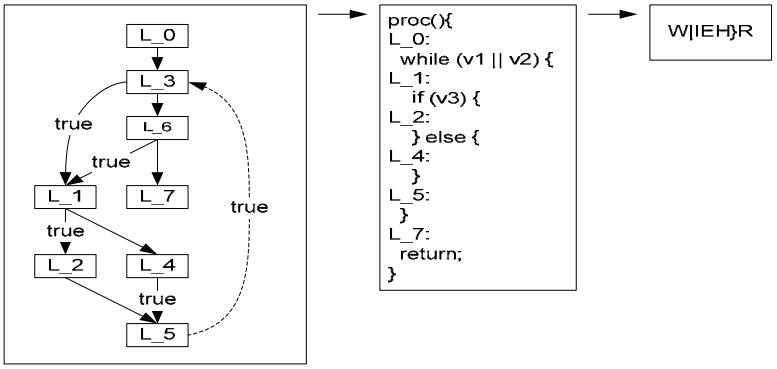
Our approach: A set of control flow graphs birthmark

- Intra-procedural control flow.
- Many procedures.



TRANSFORMING GRAPH DISSIMILARITY TO A STRING DISSIMILARITY PROBLEM

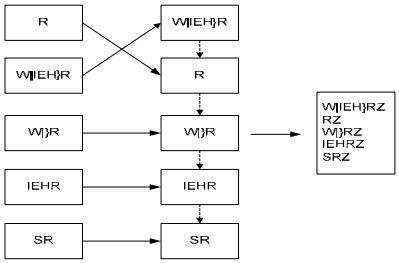
- Decompile control flow graphs to strings.
- Compare strings using 'string metrics'.



NEW APPROACHES TO FLOWGRAPH-BASED CLASSIFICATION

TRANSFORMING A SET OF STRINGS PROBLEM INTO A STRING PROBLEM

- Decompiled CFGs give us a set of strings.
- Order and concatenate strings.
- Deliminate substrings with 'Z'.
- Order based on metrics.
  - Number of instructions in procedure.
  - Number of basic blocks.
  - etc



WHAT WE TRIED (AND ENDED UP NOT USING)

- String metrics:
  - Edit distance → ed("h**ello**", "gg**ello**") = 2
  - Normalized Compression Distance → $NCD(x,y) = \frac{C(xy) - \min\{C(x), (C,y)\}}{\max\{C(x), C(y)\}}$
  - Sequence alignment →

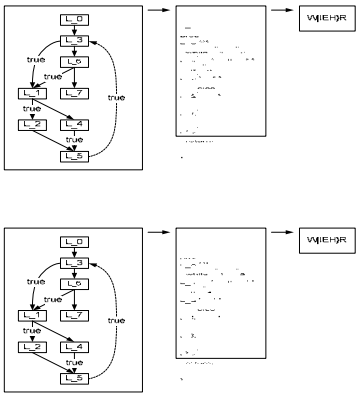
```
A K TKT K
| | | | |
ATKTT T K
```

- All databases indexed using metric trees.

SEQUENCE ALIGNMENT WITH BLAST

- A heuristic genome sequence search tool.
- Local sequence alignment.
- Hugely popular in bioinformatics.
- So.. transform our strings into genome sequences.
- Then, do a genome search.

GENOME SEQUENCE EXTRACTION



→ ACGTRYKMACGTRYKM

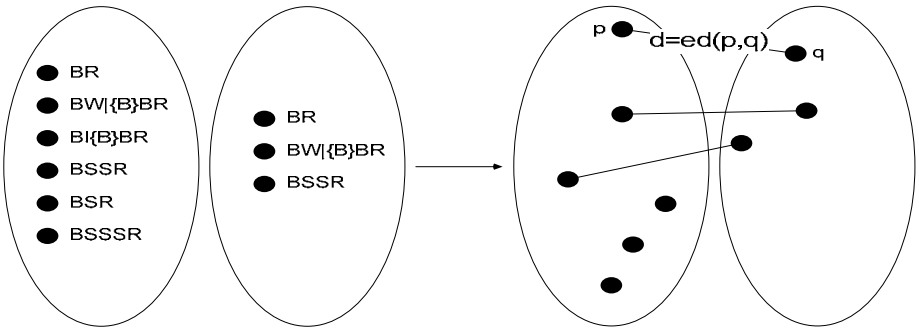
A = Adeline
C = Cytosine
G = Guanine
T = Thyamine
...

WHY DIDN'T WE USE THOSE APPROACHES?

- Not optimally effective.
- Too slow.
- Best speed was using NCD.

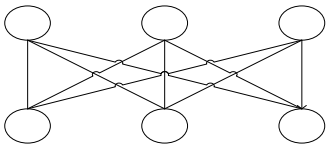
A DISSIMILARITY METRIC FOR SETS OF STRINGS (WHAT WE ENDED UP USING)

- Find a mapping between strings to minimize the sum of distances.



COMBINATORIAL OPTIMISATION: THE ASSIGNMENT PROBLEM

- Finding a minimum cost mapping is known as the "assignment problem"
- Optimal solutions exist in cubic time.
- "Greedy" heuristic solutions faster.
- Has the properties of a metric.



EVALUATION

IMPLEMENTATION

- Malwise system is 100,000 lines of code of C++.
- The modules for this work < 3000 lines of code.
- Unpacks malware using an application level emulator (Ruxcon 2010)
- Pre-filtering stage to quickly cull non matching variants (Ruxcon 2011)

EVALUATION - EFFECTIVENESS

- Calculated similarity between Roron malware variants.
- Compared results to Ruxcon 2010 work.
- In tables, highlighted cells indicates a positive match.
- The more matches the more effective it is.

Evaluation - Effectiveness

	ao	b	d	e	g	k	m	q	a
ao		0.44	0.28	0.27	0.28	0.55	0.44	0.44	0.47
b	0.44		0.27	0.27	0.27	0.51	1.00	1.00	0.58
d	0.28	0.27		0.48	0.56	0.27	0.27	0.27	0.27
e	0.27	0.27	0.48		0.59	0.27	0.27	0.27	0.27
g	0.28	0.27	0.56	0.59		0.27	0.27	0.27	0.27
k	0.55	0.51	0.27	0.27	0.27		0.51	0.51	0.75
m	0.44	1.00	0.27	0.27	0.27	0.51		1.00	0.58
q	0.44	1.00	0.27	0.27	0.27	0.51	1.00		0.58
a	0.47	0.58	0.27	0.27	0.27	0.75	0.58	0.58	

Exact Matching (Ruxcon 2010)

	ao	b	d	e	g	k	m	q	a
ao		0.70	0.28	0.28	0.27	0.75	0.70	0.70	0.75
b	0.74		0.31	0.34	0.33	0.82	1.00	1.00	0.87
d	0.28	0.29		0.50	0.74	0.29	0.29	0.29	0.29
e	0.31	0.34	0.50		0.64	0.32	0.34	0.34	0.33
g	0.27	0.33	0.74	0.64		0.29	0.33	0.33	0.30
k	0.75	0.82	0.29	0.30	0.29		0.82	0.82	0.96
m	0.74	1.00	0.31	0.34	0.33	0.82		1.00	0.87
q	0.74	1.00	0.31	0.34	0.33	0.82	1.00		0.87
a	0.75	0.87	0.30	0.31	0.30	0.96	0.87	0.87	

Heuristic Approximate Matching (Ruxcon 2010)

	ao	b	d	e	g	k	m	q	a
ao		0.86	0.49	0.54	0.50	0.87	0.86	0.86	0.86
b	0.87		0.57	0.63	0.62	0.96	1.00	1.00	0.96
d	0.61	0.64		0.85	0.91	0.64	0.64	0.64	0.64
e	0.64	0.69	0.85		0.90	0.68	0.69	0.69	0.68
g	0.62	0.68	0.91	0.91		0.68	0.68	0.68	0.68
k	0.88	0.96	0.58	0.62	0.61		0.96	0.96	0.99
m	0.87	1.00	0.57	0.63	0.62	0.96		1.00	0.96
q	0.87	1.00	0.57	0.63	0.62	0.96	1.00		0.96
a	0.87	0.96	0.58	0.62	0.61	0.99	0.96	0.96	

Assignment problem

Evaluation – False Positives

- Database of 10,000 malware.
- Scanned 1,601 benign binaries.
- 7 false positives. Less than 1%.
- Very small binaries have small signatures and cause weak matching.

Evaluation - Efficiency

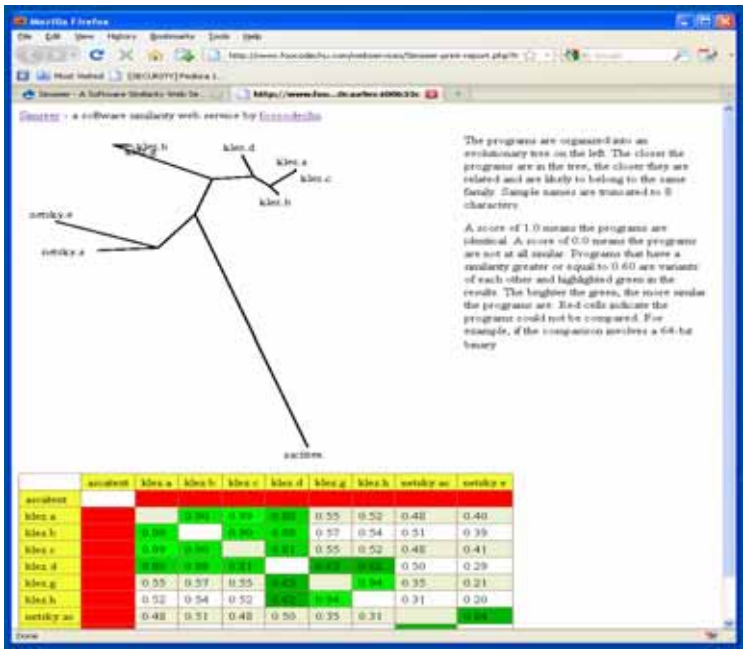
- Median benign and malware processing time is 0.06s and 0.84s.

% Samples	Benign Time(s)	Malware Time(s)
10	0.02	0.16
20	0.02	0.28
30	0.03	0.30
40	0.03	0.36
50	0.06	0.84
60	0.09	0.94
70	0.13	0.97
80	0.25	1.03
90	0.56	1.31
100	8.06	585.16

But that's not all we use the Malwise engine for..

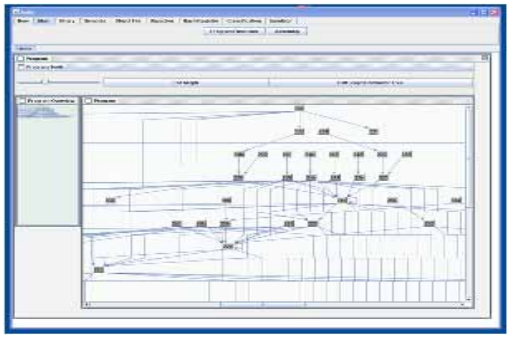
SIMSEER – A SOFTWARE SIMILARITY WEB SERVICE

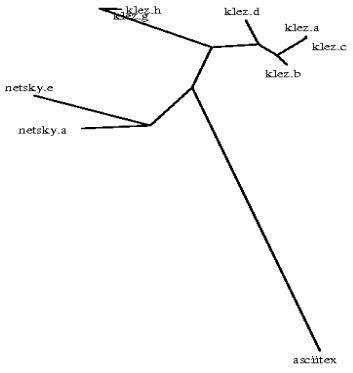
- An online service to identify similarity between programs
- Based on Malwise.
- Renders an evolutionary tree to show program relationships.
- Free to use!
- http://www.foocodechu.com/?q=simseer-a-software-similarity-web-service



SIMSEER - DEMO

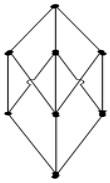
- http://www.youtube.com/watch?v=ymo7DKlKCH4





BUGWISE

- Automatically detect bugs and vulnerabilities in Linux executable binaries.
- Uses static program analysis from Malwise.
  - Decompilation
  - Data Flow Analysis →
- Free to use!
- http://www.foocodechu.com/?q=bugwise-a-bug-detection-web-service-for-binary-executables



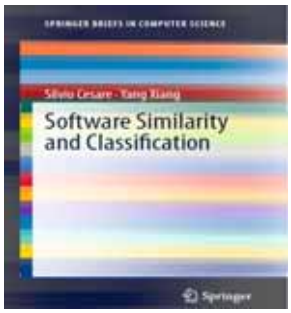
BUGWISE – SGID GAMES XONIX BUG IN DEBIAN LINUX

```
memset(score_rec[i].login, 0, 11);
strncpy(score_rec[i].login, pw->pw_name, 10);
memset(score_rec[i].full, 0, 65);
strncpy(score_rec[i].full, fullname, 64);
score_rec[i].tstamp = time(NULL);
free(fullname);

if((high = freopen(PATH_HIGHSCORE, "w",high)) == NULL) {
  fprintf(stderr, "xonix: cannot reopen high score file\n");
  free(fullname);
  gameover_pending = 0;
  return;
}
```

PUBLICATIONS

- Book published by Springer.
- http://www.springer.com/computer/security+and+cryptology/book/978-1-4471-2908-0



CONCLUSION

- Malwise effectively identifies malware variants.
- Runs in real-time in expected case.
- Large functional code base and years of development time.
- Happy to talk to vendors.
- http://www.FooCodeChu.com